Amendments to the Claims

The listing of claims will replace all prior versions, and listings of claims in the application.

1. (*currently amended*) A method for supporting development of content independent of a run time platform, comprising the steps of:

storing processing blocks that define content; and

storing an application graph that expresses the identity of the stored processing blocks and data connectivity between the stored processing blocks; ~~whereby,~~

wherein the application graph can be traversed by a graphical application platform at run time to execute appropriate processing blocks on a run time platform, and

wherein an execution of at least one of the appropriate processing blocks can cause the application graph to be modified by the graphical application platform.

2. (*original*) The method of claim 1, wherein the content comprises game content.

3. (*currently amended*) A method for supporting development of content independent of multiple hardware platforms, comprising the steps of:

storing processing blocks that define content independent of multiple hardware platforms;

selecting a target hardware platform from multiple hardware platforms;

storing an application graph that expresses the identity of the stored processing blocks and data connectivity between the stored processing blocks based on the selected target hardware platform; and

traversing the application graph at run time, including executing appropriate processing blocks on the selected target hardware platform, and modifying the application graph as a result of executing at least one of the appropriate processing blocks.

4. (*original*) The method of claim 3, wherein the content comprises game content, and the multiple hardware platforms include at least one of a game console platform and a personal computer platform.

5. (*currently amended*) A game development and run time system, comprising:

a graphical application platform that enables a game application to run on any of multiple hardware platforms, said graphical application platform comprising:

an application real time kernel,

a plurality of standard features implemented as executable blocks of logic, and

connections between said blocks that implement data flow between said blocks such that capabilities of the game application and any of the multiple hardware platforms can be implemented modularly by adding additional corresponding blocks and connections,

wherein the application real time kernel can modify ~~modifies~~ the connections between said blocks at run time as a result of executing at least one of said blocks.

6. (*original*) The system of claim 5, further comprising:

an object definition tool that enables a developer to define an application graph such that said game application can run on a target hardware platform.

7. (*original*) The system of claim 6, wherein said object definition tool further enables a developer to define objects, object elements, and connections.

8. (*currently amended*) A graphical application platform for leveraging capabilities provided independently in at least one of an application software and a hardware platform, comprising:

an application real time kernel (ARK);

a plurality of standard features implemented as executable blocks of logic; and

connections between said blocks that implement data flow between said blocks, whereby capabilities of at least one of the application software and the hardware platform can be implemented modularly by adding additional corresponding blocks and connections,

wherein the application real time kernel can modify ~~modifies~~ the connections between said blocks at run time as a result of executing at least one of said blocks.

9. (*original*) The graphical application platform of claim 8, wherein said ARK comprises logic that invokes blocks according to a schedule listing the blocks to be executed in each of at least one ARK thread running on at least one central processing unit, dynamically loads and unloads blocks, monitors block execution, and facilitates thread management, memory sharing, mutual exclusion, and synchronization.

10. (*original*) The graphical application platform of claim 8, wherein said additional blocks implement additional features, said additional features comprising market oriented features.

11. (*original*) The graphical application platform of claim 8, wherein said additional blocks implement additional features, said additional features comprising application specific features.

12. (*original*) The graphical application platform of claim 8, wherein said standard and additional blocks are organized into components, wherein each component comprises blocks representing alternative implementations of a feature.

13. (*original*) The graphical application platform of claim 12, wherein each of said alternative implementations comprises:

a) blocks corresponding to said alternative implementation;

b) identification of resources needed by said alternative implementation; and

c) identification of resources provided by said alternative implementation.

14. (*currently amended*) A method of ~~pre-processing~~ executing a feature for a graphics application with respect to a predefined hardware platform, comprising the steps of:

a) selecting from among a set of alternative implementations of a feature;

b) mapping at least one block, corresponding to the selected implementation, to a phase of execution;

c) mapping the phase of execution to a stage of execution;

d) creating a block execution order list corresponding to the stage of execution; and

e) submitting the stage of execution to an application real time kernel for management of execution of the stage, wherein the execution of the stage may cause an additional feature to be executed following the steps of (a)-(e).

15. (*original*) The method of claim 14, wherein said step a) comprises a negotiation process in which resource requirements of each alternative implementation are considered, along with the costs and benefits of variations in such resource requirements, thereby allowing selection of an implementation.

16. (*currently amended*) A system for supporting development of content independent of a run time platform, comprising:

means for storing processing blocks that define content; and

means for storing an application graph that expresses the identity of the stored processing blocks and data connectivity between the stored processing blocks; ~~whereby,~~

wherein the application graph can be traversed by a graphical application platform at run time to execute appropriate processing blocks on a run time platform, and

wherein an execution of at least one of the appropriate processing blocks can cause the application graph to be modified by the graphical application platform.

17. (*previously presented*) The system of claim 16, wherein the content comprises game content.

18. (*currently amended*) A system for supporting development of content independent of multiple hardware platforms, comprising:

means for storing processing blocks that define content independent of multiple hardware platforms;

means for enabling selection of a target hardware platform from multiple hardware platforms;

means for storing an application graph that expresses the identity of the stored processing blocks and data connectivity between the stored processing blocks based on the selected target hardware platform; and

means for traversing the application graph at run time, including executing appropriate processing blocks on the selected target hardware platform, and modifying the application graph as a result of executing at least one of the appropriate processing blocks.

19. (*previously presented*) The system of claim 18, wherein the content comprises game content, and the multiple hardware platforms include at least one of a game console platform and a personal computer platform.

20. (*currently amended*) A computer based method of dynamically configuring the execution of an application for a particular hardware configuration comprising:

loading an application graph of the application on a hardware configuration, wherein the application graph is a directed data flow graph wherein a node of the application graph identifies a processing block and links of the application graph indicate the direction of the data flow between processing blocks, wherein the application graph is independent of hardware configurations;

loading a dictionary of components on the hardware configuration, wherein a component represents a feature, wherein a component contains at least one implementation of the feature, wherein an implementation of the feature is represented by a set of processing blocks, wherein an implementation of the feature is customized for at least one hardware configuration;

inserting at least one set of processing blocks representing an implementation of a component into the application graph, wherein the implementation is customized for the hardware configuration; and

executing the application graph, wherein the application graph is dynamically modified, wherein said executing step includes:

traversing the application graph;

executing the processing blocks of the application graph, wherein executing the processing blocks specifies a necessary feature;

selecting a component from the dictionary based on the specified necessary feature;

selecting an implementation from the selected component based on the hardware configuration; and

modifying the application graph dynamically at run time by inserting the set of processing blocks of the selected implementation into the application graph.

21. (*previously presented*) The computer based method of claim 20, wherein the traversing step segregates the processing blocks into threads, wherein the traversing step comprises:

mapping each processing block in the application graph to one of a plurality of phase of executions, wherein a phase of execution represents a subset of the

application graph wherein all processing blocks in one phase of execution can be executed before or after the execution of another phase of execution;

mapping each phase of execution to one of a plurality of stage of executions, wherein all processing blocks in all phases of execution in a stage of execution can run on a single thread;

creating a block execution order list for each stage of execution, wherein a block execution order list indicates the order in which each processing block in a stage of execution will be executed; and

assigning each stage of execution to one of a plurality of threads, wherein all the processing blocks of a stage of execution will execute on the assigned thread according to the block execution order list for the stage of execution.

22. (*previously presented*) The computer based method of claim 21, wherein said creating step comprises:

creating the block execution order list for each stage of execution by ordering the processing blocks according to the frequency in which each processing block should execute.

23. (*previously presented*) The computer based method of claim 20, wherein each implementation of a feature includes a resource negotiation information, and wherein said selecting an implementation step includes:

identifying a set of implementations in the selected component that is compatible with the hardware configuration;

identifying the resources available on the hardware configuration; and

selecting the implementation from the set of implementations based on the resource negotiation information for each implementation in the set of implementations and the resources available on the hardware configuration.